N88-16413

Commonality Analysis as a Knowledge Acquisition Problem

Dorian P. Yeager
The University of Alabama
College of Engineering
Department of Computer Science
Tuscaloosa, Alabama 35487

Abstract. Commonality Analysis is a systematic attempt to reduce costs in a large scale engineering project by discontinuing development of certain components during the design phase. Each discontinued component is replaced by another component which has sufficient functionality to be considered an appropriate substitute. The replacement strategy is driven by economic considerations. The tool currently in use by NASA to guide the commonalty analysis process, known as the System Commonality Analysis Tool (SCAT), is based on an oversimplified model of the problem and incorporates no knowledge acquisition component. In fact, the process of arriving at a compromise between functionality and economy is quite complex, with many opportunities for the application of expert knowledge. Such knowledge is of two types: (1) general knowledge expressable as heuristics and mathematical laws potentially applicable to any set of components, and (2) specific knowledge about the way in which elements of a given set of components interrelate. Examples of both types of knowledge are presented, and a framework is proposed for integrating the knowledge into a more general and useable tool.

Introduction. Component part standardization has been used as a means of increasing volume and reducing the cost of manufacturing goods since the industrial revolution. The major cost saving was due to mass production, which dramatically reduced the cost of producing each unit. A side benefit was that items manufactured in this way were cheaper and easier to repair, because replacement parts were plentiful and reliable. Commonality is a similar technique, applied at a higher level. Commonality analysis attempts to standardize components on a system-wide basis, or across multiple systems in a large engineering effort. The components involved are more complex, serving multiple functions. For example, Boeing Corporation has saved millions of dollars in development, production, and maintenance costs, as well as in pilot training, by employing identical cockpits in the Boeing 757 and 767 aircraft. The earlier in a large engineering effort that the principle of commonality is employed, the greater the potential cost-saving benefits.

In a general sense, commonality analysis refers to an objective evaluation of a large and complex project at a fairly early stage in its design with the goal of finding opportunities to apply the principle of commonality. Much of what can be called commonality analysis is highly creative and has no fixed methodology. However, there is one activity that appears to run through all such analyses as a unifying thread: the direct comparison of two or more competing designs, to ascertain the feasibility of eliminating some of those designs. Often the functionality of an item can be extended in such a way that it may serve other purposes while continuing to function in the original fashion as well. It may also be possible to use multiple copies of one item in place of another. In still other cases it may be possible to make a simple substitution, eliminating an item whose functionality can be completely assumed by another.

Current Software Solutions. Only one type of comparison lends itself easily to automation via software, and that is the type of comparison which strives to evaluate the advisability of substituting one or more copies of one item for another, without examining the possibility of redesigning or extending the functionality of any items. In this case it is often sufficient to simply evaluate a cost function. In the case of a two-way comparison, say between item a and item b, the cost function must be evaluated for three situations: that in which a and b are uniquely implemented, that in which a substitutes for b, and that in which b substitutes for a. The three numbers are compared, and the lower cost wins. This simple strategy is the basis for a software tool currently in use by NASA, called the System Commonality Analysis Tool, or SCAT (See [1]). SCAT evaluates a set of n objects by computing n+1 costs: the so-called "unique option", plus the n possible substitution strategies.

Of course, cost functions must be given sufficient data on each item in order to give realistic predictions of comparative costs. The gathering and management of that data is another need which indicates a software solution. The SCAT program incorporates data management facilities. In fact, SCAT is written as a front end to a commercial database management system (DBMS). SCAT obtains the data it needs for its cost analyses from files created with the DBMS functions.

SCAT operates as follows. Design data on hardware and/or software components are captured as records in commonality databases. Each such database is created and maintained by a database administrator familiar with the project. A separate record is made for each item which may be a candidate for comparative cost analysis. The attributes of a record must always include those required by the SCAT cost function. To insure this, the database administrator is constrained to create the database via the SCAT front end, which automatically supplies the needed attributes with each new database. However, there is no requirement that all items entered into a database have identical, or even very similar, functional characteristics. Nor is there any capability within SCAT to search for sets of items with related functional characteristics. For its comparative cost analyses, SCAT relies on the database administrator to communicate to it precisely the subset of n items which it is to evaluate. This is done with standard database subsetting operations, communicated via a series of menus which painstakingly prompt for the necessary information to construct a relational expression to be used as a query. For those with more relational database experience, direct access to the DBMS proper is provided.

Once the subset database is identified which is to be subjected to analysis, the SCAT user may request a cost analysis on that subset. SCAT then assumes that the items in the subset have identical functionality, and provides the requested n+1 cost figures, sorted in increasing order. The final assumption is that the most "cost effective" alternative will be a viable alternative.

A More General Formulation. Let α be the relation, defined on the set of all records in a given commonality database, as $a\alpha b$ if and only if a is a feasible substitute for b. We call α the feasibility relation on that set of records. The properties of the relation α depend entirely on the characteristics of the given database. α may or may not be symmetric, antisymmetric, or transitive. As a convention, we take $a\alpha a$ to be true for all items a in the database (that is, α is always reflexive). By rights, it is the connected components of the relation α that ought to be subjected to analysis. In other words, if a record x is in a given subset being subjected to analysis, we would wish that all records y be also in the subset, where there is a series of records x_1, x_2, ..., x_m, for which $x_1 = x$, $x_m = y$, and for each i = 1, 2, ..., m-1, either $x_i \alpha x_{i+1}$ or $x_{i+1} \alpha x_i$.

Let us assume that we have isolated one of these subsets, say A, and that it is in fact a connected component of α. The form of the relation on set A may be arbitrarily complex. Let us consider the simple case of a two-element set, the two elements a and b we referred to in our discussion of SCAT, above. If both $a\alpha b$ and $b\alpha a$ are true, then all three SCAT options make sense, and we choose the least costly. If only one of them is true, for example if $a\alpha b$ and not $b\alpha a$, then we may or may not choose to replace b by a, even though α permits us to do so. It may be more cost-effective to produce the two items separately. However, if we run a SCAT analysis on the set, the recommendation may be to substitute b for a, even though that is not a viable alternative. SCAT's recommendations must be filtered through a human expert, who knows which solutions make sense and which do not. Now let us add a third element, c, to the set. An interesting fact here is that the most economical alternative may be to substitute a for c and produce b uniquely. This may be true because of the form of the relation α. For example, it may be that the only two non-reflexive relationships are $a\alpha c$ and $b\alpha c$. However, depending on which cost function one uses, such a twofold strategy may be called for even if α freely allows substitutions of all kinds in the set $\{a,b,c\}$.

The most general substitution strategy is represented by a pair (π,T), where π is a partition of the set A and T is a set of representatives of π. In the example above, the partition is π =

$\{\{a,c\},\{b\}\}$ and the set of representatives is $T = \{a,b\}$. If $\pi = \{K_1, K_2, \ldots, K_m\}$, and $T = \{t_1, t_2, \ldots, t_m\}$, then it must be true that for each $i = 1, 2, \ldots, m$, $t_i \alpha x$ for all x in K_i. For this reason we call (π,T) an α-partition.

A SCAT-type solution can now be seen as a special case of this general form. It is the case where the partition π and the set T have only one element each. That is, $\pi = \{A\}$ and $T = \{t\}$ for some element t of A.

The Need for a New Methodology. Clearly, techniques for generating the more general form of solution described above will be much more complex than the simple SCAT strategy. An initial collection of knowledge about α-partitions and cost functions on α-partitions is available in the form of a series of propositions contained in a paper [2] submitted by the author to the journal, Operations Research. Several of these propositions suggest algorithms which may be applied to provide a sub-optimal solution, which may then be refined by heuristic techniques. Because of the very general nature of the problem, there probably is no deterministic algorithm which will yield an optimal solution in every case, and each case must be examined in light of its own properties. An eclectic solution strategy is called for. Logic programming is the obvious tool for investigating such solution strategies because of the natural way in which propositional knowledge may be encoded.

Capturing the Feasibility Relation. The perfecting of a generalized solution strategy for commonality analysis is an intriguing problem, but there is a companion problem which is just as intriguing. To be able to say that widget a is a feasible substitute for widget b clearly requires expert knowledge about widgets. To search through a database of hundreds of widget designs and produce a set of twelve which are closely related to the extent that a SCAT-type analysis may be performed on that set also requires a certain level of expertise. Is there any hope that this process may yield to a software solution? If so, then a knowledge base component is necessary.

It is possible to capture the knowledge about α and store it as an integral part of the commonality database itself. Clearly, there must be a close physical association between the data and the knowledge whereby the relation α on that data may be constructed. We propose, then, that every commonality database be accompanied by a companion knowledge base. The construction and maintenance of the knowledge base would be the responsibility of the database administrator.

Let us examine how the knowledge might be encoded. In the SCAT environment, the user is encouraged to find a set of items for analysis by sorting on various attributes and scanning the sorted list for potentially common sets of items. When such a group appears, the user may communicate to SCAT the set he or she is interested in by means of a relational expression which identifies the desired set. If the decisions concerning how to sort and group the data are made in advance, the entire process of selecting a subset for analysis can be carried out in a single automated operation.

But let us not confine ourselves to SCAT-type methodology. What we are trying to do is to capture the feasibility relation α. Any information about α will be useful, even if it consists only of a single pair (a,b) of records. The forms taken by the knowledge will be varied. The following list covers some of those forms.

Type of Information	Parameters Needed
pair	<record key>_1, <record key>_2
sort	<attribute>,<direction>,...
group	<attribute>,<range of values> <attribute>,<relation> <attribute>,<tolerance>
relational expressions	No specific form for parameters. May use a specially designed prefix or postfix coding scheme.

Conclusions. The Commonality Analysis problem requires expert knowledge at all phases of the solution process. The creation of databases, the maintenance of data and knowledge about the data, the selection of commonality alternatives, and the application of solution strategies may all profit from software solutions that incorporate knowledge. The report [3] referenced below presents an overall strategy for the incorporation of knowledge.

Acknowledgements. The author expresses his gratitude to NASA and the NASA/ASEE Summer Faculty Fellowship program, which sponsored his initial research into this topic. Special thanks go to Dale Thomas for his enthusiasm, encouragement, and ideas.

References.

1. MSFC. Commonality Analysis Study, User Manual for the System Commonality Analysis Tool (SCAT), D483-10064, March 1987. Contract NAS8-36413, NASA George C. Marshall Space Flight Center, Alabama.

2. Yeager, D. P. A Formulation of the Commonality Analysis Problem and Some Partial Solutions, submitted to <u>Operations Research</u>, 1987.

3. Yeager, D. P. Expert System Development for Commonality Analysis in Space Programs, in final report of the 1987 NASA/ASEE Summer Faculty Fellowship Program, NASA George C. Marshall Space Flight Center, pp. XXXV-i through XXXV-25.